Irregular Product Coded Computation for High-Dimensional Matrix Multiplication

Hyegyeong Park and Jaekyun Moon
School of Electrical Engineering
Korea Advanced Institute of Science and Technology (KAIST)
Email: parkh@kaist.ac.kr, jmoon@kaist.edu

***Abstract*—In this paper, we consider the straggler problem of the high-dimensional matrix multiplication over distributed workers. To tackle this problem, we propose an irregular-product-coded computation, which is a generalized scheme of the standard-product-coded computation proposed in [1]. Introducing the *irregularity* to the product-coded matrix multiplication, one can further speed up the matrix multiplication, enjoying the low decoding complexity of the product code. The idea behind the irregular product code introduced in [2] is allowing different code rates for the row and column constituent codes of the product code. We provide a latency analysis of the proposed irregular-product-coded computation. In terms of the total execution time, which is defined by a function of the computation time and decoding time, it is shown that the irregular-product-coded scheme outperforms other competing schemes including the replication, MDS-coded and standard-product-coded schemes in a specific regime.**

I. Introduction

In the era of Big data, the massive scale of data often impede running computations over a single machine within a reasonable computation time. To overcome this limitation, distributed computing systems have played a pivotal role in supporting the large-scale data analytics in recent years [3]. In a distributed computing system a computational task is divided into a number of subtasks and distributed over multiple workers. The computationally intensive task can then benefit from parallelism, which leads to reducing the overall computation time.

With the growth in the scale of data/systems, however, the distributed computing system often faces a serious bottleneck in performance to wait some straggling workers or *stragglers* [4]. To provide the straggler tolerance to the distributed system, coding for distributed matrix multiplication is first introduced in [5]. In [5], it is shown that deploying an (n, k) maximum distance separable (MDS) code speeds up the distributed linear computations including the matrix multiplication since the master can retrieve the computation result by collecting any k out of n workers' subtask result.

Coding for distributed computing is a very active area of research. A number of recent publications have explored the role of coding in distributed computing in the following various contexts: coding for high-dimensional matrix multiplication [1], [6], [7], distributed gradient descent [8]–[11], Fourier transform [12], convolution [13] and non-linear computation [14]. Moreover, there have been works to reflect the practical constraint of distributed computing systems: coding for matrix-vector multiplication over heterogeneous clusters [15]–[17] and hierarchical architectures [18], [19]. Another line of work deals with the exploitation of stragglers' computation results [20]–[22].

The most relevant past work is [1], which considers two different regimes that the number of backup workers either sublinearly or linearly scales in the code dimension (or, in the size of the product, equivalently). The authors compare the computation time of the MDS-coded and product-coded schemes in those two regimes. It is shown that the MDS-coded computation outperforms the product-coded computation in the regime of linearly scaling number of backup workers, while the product code has a better performance in another regime. However, we claim that the product-coded computation has a room for improvement since the design of the standard product code employed in [1] has not been optimized to have a lower latency.

This motivates us to explore the impact of proper designs of the product code to further speedup the matrix multiplication. Moreover, the product code can take advantage in decoding time over the MDS code as the product code consists of small constituent codes, which enables the efficient parallel decoding of small codes. Improving the computation time of the product code will then greatly help to reduce the latency in distributed matrix multiplication in terms of the total execution time, i.e., the summation of the computation time and decoding time.

For improving the latency of the product code, we specifically consider the irregular product code [2]. Irregular product codes can achieve a better performance compared to the standard product codes by allowing each row and column constituent code to have different code rates. This is because a few lower rate constituent codes enhances the straggler tolerance, while the higher rate constituent codes force the irregular product code to maintain the overall code rate. In this paper, we employ the irregular product code for distributed matrix multiplication and show that the irregular-product-coded scheme outperforms the MDS-coded scheme in some specific regime.

Target Problem: We consider matrix multiplication over N distributed workers, i.e., we aim to compute $\boldsymbol{A}^T\boldsymbol{B}$ for $\boldsymbol{A} \in \mathbb{R}^{s\times d}$ and $\boldsymbol{B} \in \mathbb{R}^{s\times t}$. The size of the input matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ is assumed to be fairly large; $\boldsymbol{A}$ and $\boldsymbol{B}$ are thus required to be split into small chunks to avoid computations

that are physically not feasible in a single machine. The input matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are first divided into, for example, k blocks as $\boldsymbol{A} = [\boldsymbol{A}_1 \ \boldsymbol{A}_2 \ \dots \ \boldsymbol{A}_k]$ and $\boldsymbol{B} = [\boldsymbol{B}_1 \ \boldsymbol{B}_2 \ \dots \ \boldsymbol{B}_k]$, respectively, where $\boldsymbol{A}_i \in \mathbb{R}^{s \times \frac{d}{k}}$ and $\boldsymbol{B}_j \in \mathbb{R}^{s \times \frac{t}{k}}$ for $i, j \in [k]$. Here d and t are assumed to be divisible by k for simplicity. Then the submatrices $\boldsymbol{A}_1, \boldsymbol{A}_2, \dots, \boldsymbol{A}_k$ and $\boldsymbol{B}_1, \boldsymbol{B}_2, \dots, \boldsymbol{B}_k$ are mapped into the coded submatrices $\hat{\boldsymbol{A}}_1, \hat{\boldsymbol{A}}_2, \dots, \hat{\boldsymbol{A}}_N$ and $\hat{\boldsymbol{B}}_1, \hat{\boldsymbol{B}}_2, \dots, \hat{\boldsymbol{B}}_N$, respectively. Worker w is assigned a subtask to compute $\hat{\boldsymbol{A}}_w^T \hat{\boldsymbol{B}}_w$, and the master can retrieve the desired computation result $\boldsymbol{A}^T\boldsymbol{B}$ by gathering only a subset of $\{\hat{\boldsymbol{A}}_w^T \hat{\boldsymbol{B}}_w\}_{w \in [N]}$.

We want to find a proper design for mapping functions from the block of input matrices $\boldsymbol{A}_1, \boldsymbol{A}_2, \dots, \boldsymbol{A}_k$ and $\boldsymbol{B}_1, \boldsymbol{B}_2, \dots, \boldsymbol{B}_k$ to the coded matrix blocks $\hat{\boldsymbol{A}}_1, \hat{\boldsymbol{A}}_2, \dots, \hat{\boldsymbol{A}}_N$ and $\hat{\boldsymbol{B}}_1, \hat{\boldsymbol{B}}_2, \dots, \hat{\boldsymbol{B}}_N$, respectively. One possible direction is a design for minimizing the total execution time which is a function of the computation and decoding time. Inspired by the decoding efficiency of the product code and its potential to improve the computation time, we specifically consider the irregular-product-coded computation. Our main goal is to analyze the latency of such irregular-product-coded computation.

Contributions: The main contributions of this paper are summarized as follows.

- We propose a novel coded computation scheme based on irregular product codes.
- We provide an analysis on the expected computation time of our proposed coded computation scheme.
- We show that in a certain regime, the irregular-product-coded computation has a better performance in terms of the total execution time, defined as a function of the computation time and decoding time, compared to the other existing schemes.

Notations: We denote $[n] = \{1, 2, \dots, n\}$ for $n \in \mathbb{N}$. The n-th harmonic number is denoted as $H_n = \sum_{i=1}^{n} 1/i$ for $n \in \mathbb{N}$. We define $H_0 \coloneqq 0$ for ease of presentation. The cardinality of set S is denoted by $|S|$. We denote the completion time of worker $w(i,j)$ by $T_{i,j}$. The number of nonzero elements in matrix $\boldsymbol{A}$ is denoted by $\mathsf{nnz}(\boldsymbol{A})$. We denote $\mathbf{0}_{m \times n}$ an $m \times n$ all-zero matrix.

II. Irregular-Product-Coded Matrix Multiplication

In this section, we propose a coded computation based on irregular product codes. We first define the irregular product code and briefly summarize the known results on the code construction and possible code dimension. The definition of the irregular product code is stated as follows.

Definition 1. *For positive integers m, n and given sequences $\{a_i\}_{i \in [m]}$ and $\{b_j\}_{j \in [n]}$, the $\{(n, a_i)\}_{i \in [m]} \times \{(m, b_j)\}_{j \in [n]}$ irregular product code $\mathcal{C}$ is a length mn code that consists of m row codes $\{(n, a_i)\}_{i \in [m]}$ and n column codes $\{(m, b_j)\}_{j \in [n]}$. That is, the codewords of $\mathcal{C}$ are represented by the array of m rows and n columns in which row i is a codeword in the (n, a_i) code and column j is a codeword in the (m, b_j) code.*

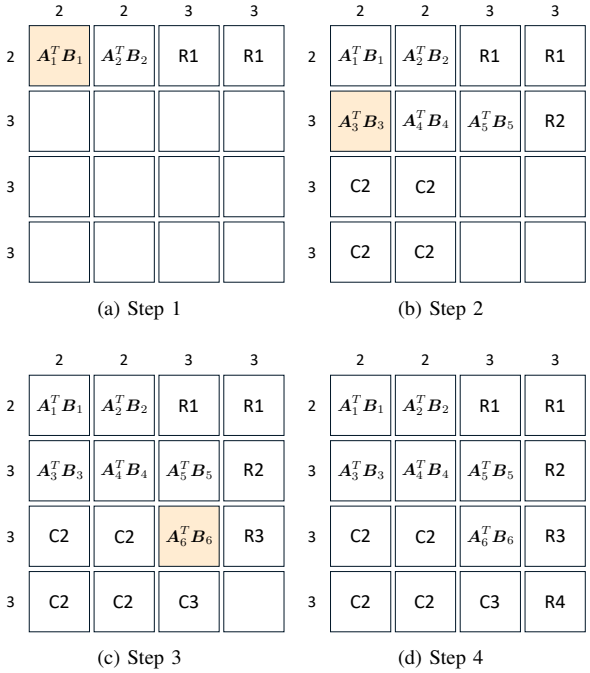


Fig. 1. Encoding of the irregular-product-coded computation

The row and column codes of the irregular product code can be any codes; however, we confine our interest to the MDS constituent (row and column) codes in this paper. Then, the following theorem provides the highest possible dimension and a corresponding achievable scheme.

Theorem 2 (From [2])**.** *Consider an $\{(n, a_i)\}_{i \in [m]} \times \{(m, b_j)\}_{j \in [n]}$ irregular product code $\mathcal{C}$. If $0 < a_1 \leq \dots \leq a_n < n$ and $0 < b_1 \leq \dots \leq b_n < m$, then the dimension of $\mathcal{C}$ is upper bounded by $k_{\mathcal{C}}$ where $k_{\mathcal{C}} \coloneqq \sum_{j=1}^{n} \sum_{i=b_{j-1}+1}^{b_j} \max(a_i - j + 1, 0)$. Furthermore, let V be the $a_m \times n$ Vandermonde matrix and V' be the $b_n \times m$ Vandermonde matrix. For $i \in [m]$ and $j \in [n]$, if the row code (n, a_i) of $\mathcal{C}$ has the first a_i rows of V as the generator matrix and the column code (m, b_j) of $\mathcal{C}$ has the first b_j rows of V' as the generator matrix, then the dimension of $\mathcal{C}$ is exactly the same as $k_{\mathcal{C}}$.*

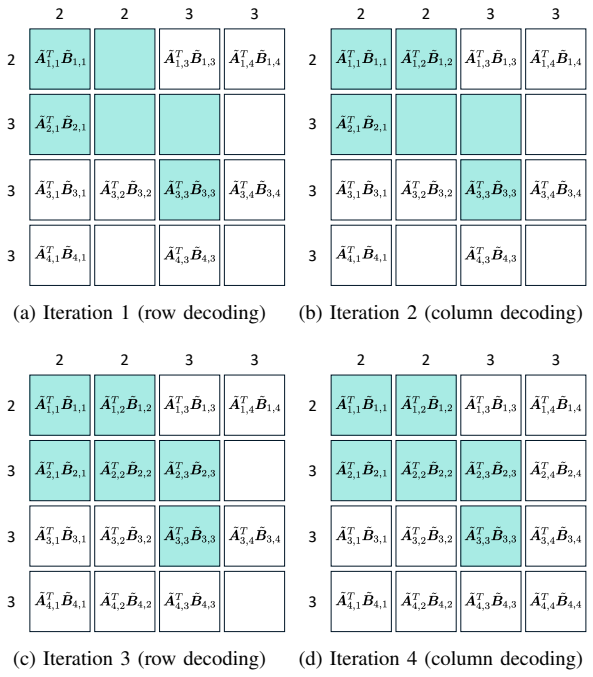
Now we present an $\{(n, a_i)\}_{i \in [m]} \times \{(m, b_j)\}_{j \in [n]}$ irregular-product-coded computation. Assume that we have mn workers, each of which is denoted by $w(i,j)$ meaning that its assigned subtask is encoded using the (n, a_i) and (m, b_j) MDS codes, i.e., i-th row and j-th column codes.

Encoding: To compute $\boldsymbol{A}^T\boldsymbol{B}$ for $\boldsymbol{A} \in \mathbb{R}^{s \times d}$ and $\boldsymbol{B} \in \mathbb{R}^{s \times t}$, the input matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are divided into $k_{\mathcal{C}}$ blocks as $\boldsymbol{A} = [\boldsymbol{A}_1 \ \boldsymbol{A}_2 \ \dots \ \boldsymbol{A}_{k_{\mathcal{C}}}]$ and $\boldsymbol{B} = [\boldsymbol{B}_1 \ \boldsymbol{B}_2 \ \dots \ \boldsymbol{B}_{k_{\mathcal{C}}}]$, respectively, where $\boldsymbol{A}_l \in \mathbb{R}^{s \times \frac{d}{k_{\mathcal{C}}}}$ and $\boldsymbol{B}_l \in \mathbb{R}^{s \times \frac{t}{k_{\mathcal{C}}}}$ for $l \in [k_{\mathcal{C}}]$. Here d and t are assumed to be divisible by $k_{\mathcal{C}}$ for simplicity. For given sequences $a_1, a_2, \dots, a_m$ and $b_1, b_2, \dots, b_n$, the procedure of encoding and assigning the submatrices to workers is described in Algorithm 1. We define the matrix $\boldsymbol{C} \in \mathbb{R}^{m \times n}$ as a data allocation matrix: $\boldsymbol{C}_{ij} \coloneqq 1$ if worker $w(i,j)$ is already assigned its submatrices; otherwise

Algorithm 1 Encoding

Input: $\{a_i, b_j\}_{i\in[m],j\in[n]}, k_{\mathcal{C}}, \{\boldsymbol{A}_l, \boldsymbol{B}_l\}_{l\in[k_{\mathcal{C}}]}$, $\boldsymbol{C} = \mathbf{0}_{m\times n}$
Set $l \leftarrow 1$ and $r \leftarrow k_{\mathcal{C}}$
while $mn - \mathsf{nnz}(\boldsymbol{C})$ **do**
  for $i \leftarrow 1$ **to** m **do**
    if $\mathsf{nnz}(i\text{-th row in } \boldsymbol{C}) = a_i$ **then**
      - Encode i-th row with an (n, a_i) MDS code
      - Assign generated parity blocks to the vacant workers of row i one by one
      - (entries with zero in the i-th row of $\boldsymbol{C}$) $\leftarrow 1$
    end if
  end for
  for $j \leftarrow 1$ **to** n **do**
    if $\mathsf{nnz}(j\text{-th column in } \boldsymbol{C}) = b_j$ **then**
      - Encode j-th column with an (m, b_j) MDS code
      - Allocate generated parity blocks to the vacant workers of column j one by one
      - (entries with zero in the j-th column of $\boldsymbol{C}$) $\leftarrow 1$
    end if
  end for
  - Find the row index i_0 and column index j_0 of the first zero element in $\boldsymbol{C}$
  if $r + \mathsf{nnz}(i_0\text{-th row in } \boldsymbol{C}) \geq a_{i_0}$ **then**
    for $u \leftarrow j_0$ **to** a_{i_0} **do**
      - Assign the task multiplying $\boldsymbol{A}_l$ and $\boldsymbol{B}_l$ to worker $w(i_0, u)$
      - $\boldsymbol{C}_{i_0,u} \leftarrow 1$, $l \leftarrow l + 1$ and $r \leftarrow r - 1$
    end for
  end if
end while
for $i \leftarrow 1$ **to** m **do**
  for $j \leftarrow 1$ **to** n **do**
    - Relabel the submatrices allocated to worker $w(i, j)$ as $\tilde{\boldsymbol{A}}_{i,j}$ and $\tilde{\boldsymbol{B}}_{j,j}$
  end for
end for
Output: $\{\tilde{\boldsymbol{A}}_{i,j}, \tilde{\boldsymbol{B}}_{i,j}\}_{i\in[m],j\in[n]}$

$\boldsymbol{C}_{ij} \coloneqq 0$. Fig. 1 shows an example of encoding procedure of the $\{(n, a_i)\}_{i\in[m]} \times \{m, b_j)\}_{j\in[n]}$ irregular-product-coded computation for given parameters $n = m = 4$, $k_{\mathcal{C}} = 6$, $[a_1, a_2, a_3, a_4] = [2, 3, 3, 3]$ and $[b_1, b_2, b_3, b_4] = [2, 2, 3, 3]$. The numbers on rows and columns denote the dimensions of each row/column. At each step, we find the row index i_0 and column index j_0 of the first zero element (in other words, the zero element with the smallest column index among the zero elements with the smallest row index) in C. The indices corresponding to the shaded areas indicate the indices i_0 and j_0 at each step. Afterwards, we assign the systematic submatrices $\boldsymbol{A}_l$ and $\boldsymbol{B}_l$ to the workers with the row index i_0 and column indices from j_0 to a_{i_0}. We then conduct the row- and column-wise encoding to generate/assign the parity blocks represented by R and C combined with the number indicating each step, respectively.



(a) Iteration 1 (row decoding) (b) Iteration 2 (column decoding)

(c) Iteration 3 (row decoding) (d) Iteration 4 (column decoding)

Fig. 2. Iterative decoding of the irregular-product-coded computation

Decoding: The irregular product code is decoded by iterative row/column code decoding using a peeling decoder. At the first iteration, for example, decoding of the m row codes is done in parallel. Specifically, for the i-th row code, decoding succeeds when any a_i out of n subtasks are completed. At the second iteration, decoding of the n column codes continues in the same way. For the j-th column code, the master can recover the desired computation result by collecting any b_j out of m subtask results. This procedure of iterative row/column decoding continues until the system converges. Fig. 2 illustrates an example of the $\{(n, a_i)\}_{i\in[m]} \times \{m, b_j)\}_{j\in[n]}$ irregular-product-coded computation for the same parameters used in Fig. 1. The shaded area indicates the systematic part.

III. Latency Analysis

For the latency analysis, we assume that the completion time of each worker is exponentially distributed with rate μ, i.e., $\Pr[T_{i,j} \leq t] = 1 - e^{-\mu t}$ for $i \in [m], j \in [n]$. This assumption is consistent with the existing literature, e.g., [1], [5], [19]. From the theory of order statistics [23], the mean of the k-th order statistics out of n exponential random variables with rate μ is given by $(H_n - H_{n-k})/\mu$. For a sufficiently large k and a fixed constant γ, the k-th harmonic number H_k is approximated by $\log k + \gamma$. We further assume that $m = n$ for simplicity; however, note that the analysis can be easily extended to the case where $m \neq n$.

Consider an $\{(n, a_i)\}_{i\in[n]} \times \{n, b_j)\}_{j\in[n]}$ irregular-product-coded computation. The following theorem provides an asymptotic lower bound for the expected computation time of the irregular-product-coded computation in the regime of our interest.

Theorem 3 (Lower Bound)**.** *For a positive fixed constant $\delta \coloneqq n^2/k_{\mathcal{C}} - 1$ and any constant $\epsilon > 0$, the expected com-*

putation time of the $\{(n,a_i)\}_{i\in[n]} \times \{n,b_j)\}_{j\in[n]}$ *irregular-product-coded computation is lower bounded as* $\mathbb{E}[T] \geq \frac{1}{\mu}\log\frac{1+\delta}{\delta-\epsilon(1+\delta)} + o(1)$ *in the limit of large* n*.*

Proof: In [2], it is shown that one can successfully decode asymptotically all the symbols of the $\{(n,a_i)\}_{i\in[n]} \times \{n,b_j)\}_{j\in[n]}$ irregular product code sent over an erasure channel with erasure probability $(n^2-k_{\mathcal{C}})/n^2 = \delta/(1+\delta)-\epsilon$ when the sequences $a_1,a_2,\ldots,a_n$ and $b_1,b_2,\ldots,b_n$ are properly chosen based on the code construction in Theorem 2. This naturally leads to a conclusion that one can construct an irregular-product-coded computation such that the full recovery at the master is asymptotically guaranteed in the existence of $n^2(\frac{\delta}{1+\delta}-\epsilon)$ stragglers as n grows to infinity. Then, the lower bound for the computation time is the $n^2(1-\frac{\delta}{1+\delta}+\epsilon)$-th order statistics of n^2 exponential random variables with rate μ. ∎

Note that the lower bound in Theorem 3 can be made arbitrarily close to the expected computation time of the $(n,k_{\mathcal{C}})$ MDS coded scheme as n grows to infinity.

The following is an asymptotic upper bound for the expected computation time of the irregular-product-coded scheme in the limit of large n.

Theorem 4 (Upper Bound)**.** *For positive fixed constants* $\{\delta_i \coloneqq n/a_i - 1\}_{i\in[n]}$ *and* $\{\delta'_j \coloneqq n/b_j - 1\}_{j\in[n]}$ *such that two integer sequences satisfy* $0 < a_1 \leq \cdots \leq a_n < n$ *and* $0 < b_1 \leq \cdots \leq b_n < n$*, the expected computation time of the* $\{(n,a_i)\}_{i\in[n]} \times \{n,b_j)\}_{j\in[n]}$ *irregular-product-coded computation is upper bounded as* $\mathbb{E}[T] \leq \frac{1}{\mu}\log\frac{1+\max\{\delta_n,\delta'_n\}}{\max\{\delta_n,\delta'_n\}} + o(1)$ *in the limit of large* n*.*

Proof: The proof generalizes the bounding technique in [1]. Assume that one starts by decoding of row constituent codes. Consider the decoding of the i-th row code, i.e., (n,a_i) MDS code. We define $t_{\mathsf{u},i} = \frac{1}{\mu}\log\frac{1+\delta_i}{\delta_i} + \alpha\sqrt{\frac{\log n}{n}}$. for some fixed constant $\alpha > 0$. The probability that decoding of the i-th row code is not completed by time $t_{\mathsf{u},i}$ is given as

$$\begin{aligned}\Pr[T_{i,j} > t_{\mathsf{u},i}] &= e^{-\mu t_{\mathsf{u},i}} = e^{-\log\frac{1+\delta_i}{\delta_i} - \mu\alpha\sqrt{\frac{\log n}{n}}} \\ &= \frac{\delta_i}{1+\delta_i} e^{-\mu\alpha\sqrt{\frac{\log n}{n}}} \\ &\simeq \frac{\delta_i}{1+\delta_i}\left(1-\mu\alpha\sqrt{\frac{\log n}{n}}\right).\end{aligned}$$

Then, for decoding the i-th row, the expected number of workers that are not completed by time $t_{\mathsf{u},i}$ is represented as $\frac{n\delta_i}{1+\delta_i}\left(1-\mu\alpha\sqrt{\frac{\log n}{n}}\right) = \delta_i\left(a_i - \mu\alpha\sqrt{\frac{a_i\log n}{1+\delta_i}}\right)$.

To bound the probability that the decoding in the i-th row fails by time $t_{\mathsf{u},i}$ (the probability that the number of unfinished workers is greater than the maximum tolerable number of stragglers), we use the Hoeffding's inequality [24]:

$$\begin{aligned}&\Pr\left[\left|\{T_{i,j} > t_{\mathsf{u},i}\}_{j\in[n]}\right| \geq \delta_i a_i\right] \\ &= \Pr\left[\left|\{T_{i,j} > t_{\mathsf{u},i}\}_{j\in[n]}\right| - \delta_i\left(a_i - \mu\alpha\sqrt{\frac{a_i\log n}{1+\delta_i}}\right) \geq \gamma\right] \\ &\leq e^{-2\gamma^2/n} = e^{-2\left(\frac{\delta_i}{1+\delta_i}\right)^2\mu^2\alpha^2\log n} = n^{-2\left(\frac{\delta_i}{1+\delta_i}\right)^2\mu^2\alpha^2},\end{aligned}$$

where $\gamma \coloneqq \delta_i\mu\alpha\sqrt{a_i\log n/(1+\delta_i)}$.

For obtaining the probability that decoding for n rows all fails by time $\max_i t_{\mathsf{u},i}$, we use the union bound:

$$\begin{aligned}\Pr\left[T > \max_i t_{\mathsf{u},i}\right] &\leq \sum_{i=1}^{n}\Pr\left[\left|\{T_{i,j} > \max_i t_{\mathsf{u},i}\}_{j\in[n]}\right| \geq \delta_i a_i\right] \\ &\leq \sum_{i=1}^{n}\Pr\left[\left|\{T_{i,j} > t_{\mathsf{u},i}\}_{j\in[n]}\right| \geq \delta_i a_i\right] \\ &\leq n\cdot n^{-2\left(\frac{\delta_n}{1+\delta_n}\right)^2\mu^2\alpha^2} = o(n^{-1}),\end{aligned}$$

where the last equality follows from the fact that α can be set to an arbitrarily large number.

Then, the expected computation time satisfies

$$\begin{aligned}\mathbb{E}[T] &\leq (1-o(n^{-1}))\max_i t_{\mathsf{u},i} + o(n^{-1})\left(\max_i t_{\mathsf{u},i} + \frac{H_{n^2}}{\mu}\right) \\ &= \frac{1}{\mu}\log\frac{1+\min_i\delta_i}{\min_i\delta_i} + o(1) \\ &= \frac{1}{\mu}\log\frac{1+\delta_n}{\delta_n} + o(1). \qquad (1)\end{aligned}$$

Similarly, we have an upper bound on the expected computation time when we decode the column codes first:

$$\begin{aligned}\mathbb{E}[T] &\leq \frac{1}{\mu}\log\frac{1+\min_j\delta'_j}{\min_j\delta'_j} + o(1) \\ &= \frac{1}{\mu}\log\frac{1+\delta'_n}{\delta'_n} + o(1). \qquad (2)\end{aligned}$$

Combining (1) and (2) completes the proof. ∎

TABLE I
COMPARISONS OF VARIOUS CODING SCHEMES (COMPARISONS ARE MADE IN THE ORDER OF MAGNITUDE)

Coding scheme	Computation time ($\mathbb{E}[T_{\mathsf{comp}}]$)	Decoding cost (T_{dec})
Replication	$k_{\mathcal{C}}H_{k_{\mathcal{C}}}/(n^2\mu)$	0
MDS	$\frac{1}{\mu}\log\left(\frac{1+\delta}{\delta}\right)$	$k_{\mathcal{C}}^{\beta}$
Standard product	$\frac{1}{\mu}\log\left(\frac{1+\delta+\sqrt{1+\delta}}{\delta}\right)$	$2\sqrt{k_{\mathcal{C}}^{\beta+1}}$
Irregular product	$\mathbb{E}[T]$	$a_n b_m^{\beta} + a_n^{\beta} b_m$

IV. COMPARISON WITH COMPETING SCHEMES

In this section, we provide performance comparisons between the irregular-product-coded matrix multiplication and other competing schemes in terms of the total latency including the computation time and decoding time.

For fair comparisons, we consider an $(n^2,k_{\mathcal{C}})$ MDS code, where $n^2 = (1+\delta)k_{\mathcal{C}}$ and $k_{\mathcal{C}}$ is given in Theorem 2. We further assume that the decoding cost of the MDS-coded computation of dimension $k_{\mathcal{C}}$ is $\mathcal{O}(k_{\mathcal{C}}^{\beta})$ for some $\beta > 1$. The assumption of $\beta > 1$ is made from the known results for the practical decoding algorithms [25], [26]. Then, the decoding complexity of the $(n,\sqrt{k_{\mathcal{C}}}) \times (n,\sqrt{k_{\mathcal{C}}})$ standard-product-coded computation is $\mathcal{O}(\sqrt{k_{\mathcal{C}}}\cdot\sqrt{k_{\mathcal{C}}}^{\beta} + \sqrt{k_{\mathcal{C}}}\cdot\sqrt{k_{\mathcal{C}}}^{\beta})$. Similarly, the decoding cost of the $\{(n,a_i)\}_{i\in[n]} \times \{n,b_j)\}_{j\in[n]}$

irregular-product-coded computation is given by $\mathcal{O}(a_{\mathsf{max}}b_{\mathsf{max}}^{\beta}+a_{\mathsf{max}}^{\beta}b_{\mathsf{max}})$. The computation time and decoding complexity of the coding schemes above are summarized in Table I.

Define the total execution time as $T_{\mathsf{exec}} := T_{\mathsf{comp}} + \lambda T_{\mathsf{dec}}$, where T_{comp} is the computation time and T_{dec} is the decoding cost. Here, λ is introduced to reflect the effect of decoding time on the total execution time, which can be interpreted as a system parameter. For comparisons, we set the code parameters as follows: $n = 8$, $k_{\mathcal{C}} = 28$, $[a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8] = [5, 5, 6, 6, 6, 6, 6, 6]$ and $[b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8] = [5, 5, 5, 5, 5, 5, 5, 5]$. The rate of the exponential runtime of a worker μ is set to 0.1, and the decoding complexity parameter β is set to 2.5.

In Fig. 3, the expected total execution times of the four coded computation schemes in Table I are presented. Although the bounds are given for our proposed scheme, we can verify that the irregular-product-coded computation outperforms the other competing schemes when λ is in the range of moderate values. When the decoding cost is negligible (small λ), we see that the lower bound of our scheme is comparable with the performance of the MDS code. Even when we consider the worst case (upper bound), the irregular-product-coded scheme consistently shows a better performance compared to both of the standard-product-coded and replication scheme. However, for large λ, which means that the decoding cost matters, the replication is the optimal choice.

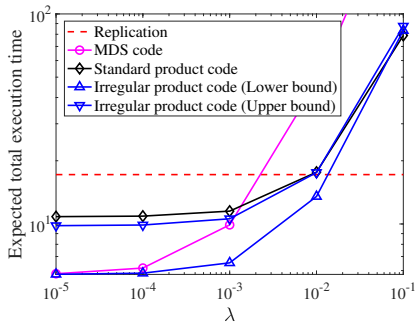


Fig. 3. Comparison of $\mathbb{E}[T_{\mathsf{exec}}]$ for various coding schemes

V. Conclusion

We proposed irregular-product-coded computing for distributed matrix multiplication. We have presented the upper and lower bounds on the latency of the proposed coded computation using the concentration inequalities. In terms of the total execution time, which is given by a function of the computation time and the decoding time, we have shown that our proposed irregular-product-coded computation outperforms the competing schemes such as the replication, MDS-coded and standard-product-coded computation, in some parameter regime. In this paper, we have focused on the product codes having MDS constituent codes. However, employing the capacity approaching constituent codes such as low-density parity-check (LDPC) codes is a promising future direction in that it may allow a lower decoding complexity at a small expense of computation time.

References

[1] K. Lee, C. Suh, and K. Ramchandran, "High-dimensional coded matrix multiplication," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2017, pp. 2418–2422.

[2] M. Alipour, O. Etesami, G. Maatouk, and A. Shokrollahi, "Irregular product codes," in *Proc. IEEE Inf. Theory Workshop (ITW)*, 2012, pp. 197–201.

[3] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le *et al.*, "Large scale distributed deep networks," in *Proc. Adv. Neural Inform. Process. Syst. (NIPS)*, 2012, pp. 1223–1231.

[4] J. Dean and L. A. Barroso, "The tail at scale," *Commun. ACM*, vol. 56, no. 2, pp. 74–80, 2013.

[5] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Trans. Inf. Theory*, vol. 64, no. 3, pp. 1514–1529, 2018.

[6] Q. Yu, M. Maddah-Ali, and S. Avestimehr, "Polynomial codes: An optimal design for high-dimensional coded matrix multiplication," in *Proc. Adv. Neural Inform. Process. Syst. (NIPS)*, 2017, pp. 4403–4413.

[7] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover, "On the optimal recovery threshold of coded matrix multiplication," *arXiv preprint arXiv:1801.10292*, 2018.

[8] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Proc. Int. Conf. Machine Learning (ICML)*, 2017, pp. 3368–3376.

[9] N. Raviv, R. Tandon, A. Dimakis, and I. Tamo, "Gradient coding from cyclic MDS codes and expander graphs," in *Proc. Int. Conf. Machine Learning (ICML)*, 2018, pp. 4305–4313.

[10] W. Halbawi, N. Azizan-Ruhi, F. Salehi, and B. Hassibi, "Improving distributed gradient descent using Reed-Solomon codes," *arXiv preprint arXiv:1706.05436*, 2017.

[11] E. Ozfatura, D. Gündüz, and S. Ulukus, "Speeding up distributed gradient descent by utilizing non-persistent stragglers," *arXiv preprint arXiv:1808.02240*, 2018.

[12] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Coded Fourier transform," in *Proc. Allerton Conf. Comm., Contr., Comp.*, 2017, pp. 494–501.

[13] S. Dutta, V. Cadambe, and P. Grover, "Coded convolution for parallel and distributed computing within a deadline," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2017, pp. 2403–2407.

[14] J. Kosaian, K. V. Rashmi, and S. Venkataraman, "Learning a code: Machine learning for approximate non-linear coded computation," *arXiv preprint arXiv:1806.01259*, 2018.

[15] A. Reisizadeh, S. Prakash, R. Pedarsani, and S. Avestimehr, "Coded computation over heterogeneous clusters," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2017, pp. 2408–2412.

[16] M. Kim, J.-y. Sohn, and J. Moon, "Coded matrix multiplication on a group-based model," *arXiv preprint arXiv:1901.05162*, 2019.

[17] D. Kim, H. Park, and J. Choi, "Optimal load allocation for coded distributed computation in heterogeneous clusters," *arXiv preprint arXiv:1904.09496*, 2019.

[18] S. Gupta and V. Lalitha, "Locality-aware hybrid coded mapreduce for server-rack architecture," in *Proc. IEEE Inf. Theory Workshop (ITW)*, 2017, pp. 459–463.

[19] H. Park, K. Lee, J. Sohn, C. Suh, and J. Moon, "Hierarchical coding for distributed computing," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2018, pp. 1630–1634.

[20] N. Ferdinand and S. C. Draper, "Hierarchical coded computation," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2018, pp. 1620–1624.

[21] S. Kiani, N. Ferdinand, and S. C. Draper, "Exploitation of stragglers in coded computation," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2018, pp. 1988–1992.

[22] A. Mallick, M. Chaudhari, and G. Joshi, "Rateless codes for near-perfect load balancing in distributed matrix-vector multiplication," *arXiv preprint arXiv:1804.10331*, 2018.

[23] H. A. David and H. N. Nagaraja, *Order Statistics*. Wiley, New York, 2003.

[24] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *J. Am. Stat. Assoc.*, vol. 58, no. 301, pp. 13–30, 1963.

[25] W. Halbawi, Z. Liu, and B. Hassibi, "Balanced Reed-Solomon codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2016, pp. 935–939.

[26] ——, "Balanced Reed-Solomon codes for all parameters," in *Proc. IEEE Inf. Theory Workshop (ITW)*, 2016, pp. 409–413.